AD-A038 231 ELECTRONIC SYSTEMS DIV HANSCOM AFB MASS F/G 9/2
MULTICS SECURITY EVALUATION. VOLUME IV. EXEMPLARY PERFORMANCE U--ETC(U)
NOV 76 G E REYNOLDS
UNCLASSIFIED ESD-TR-74-193-VOL-4 NL

| OF |
AD
A038231

END
DATE
FILMED
5-77

AD A038231

ESD-TR-74-193, Vol. IV

(12)

MULTICS SECURITY EVALUATION: EXEMPLARY PERFORMANCE UNDER DEMANDING WORKLOAD

Deputy for Command and Management Systems

November 1976

V.2
See A001120

Approved for Public Release;
Distribution Unlimited.

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

DDC
RECEIVED
APR 12 1977
A

AD No.
DDC FILE COPY

LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

This technical report has been reviewed and is approved for publication.

Grover B. Junkin

GROVER B. JUNKIN, Major, USAF
Chief, Computer Security Team
Office of Plans & Management
Air Force Data Services Center

Roger R. Schell

ROGER R. SCHELL, Lt Colonel, USAF
ADP System Security Program Manager

FOR THE COMMANDER

Frank J. Emma

FRANK J. EMMA, Colonel, USAF
Director, Computer Systems Engineering
Deputy for Command & Management Systems

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-74-193, Vol. IV
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): MULTICS SECURITY EVALUATION: EXEMPLARY PERFORMANCE UNDER DEMANDING WORKLOAD, Volume IV.
5. TYPE OF REPORT & PERIOD COVERED: Final Report, Feb 73 - Apr 76
6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): George E. Reynolds
8. CONTRACT OR GRANT NUMBER(s): IN-HOUSE
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Deputy for Command and Management Systems, Electronic Systems Division, Hanscom AFB, MA 01731
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Element 64708F, Project 6917
11. CONTROLLING OFFICE NAME AND ADDRESS: Electronic Systems Division, Hanscom AFB, MA 01731
12. REPORT DATE: November 1976
13. NUMBER OF PAGES: 41
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A
16. DISTRIBUTION STATEMENT (of this Report): Approved for Public Release; Distribution Unlimited.
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
18. SUPPLEMENTARY NOTES: This is Volume IV of a 4 Volume report: Multics Security Evaluation. The other volumes are entitled: Vol. I: Results and Recommendations; Vol. II: Vulnerability Analysis; Vol. III: Password and File Encryption Techniques
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Performance Evaluation
Multics
Computer Security
Time-Sharing
Virtual Memory
Multi-Level Systems
Operating System Workload
Privacy
Protection

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The potential of Multics was examined as a candidate system for meeting the workload demands expected to arise from upgrading the computer security capabilities at Air Force Data Services Center. Experimental program runs were made in designs to test the performance of Multics under conditions representative of extremely high computational demands issuing simultaneously from a variety of input data. Multiple runs with a WWMCCS FORTRAN benchmark program were accomplished and analyzed in the light

(continued on reverse)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Continued.

of requirements for operation in a multilevel security environment. Results were favorable to anticipated requirements. No unacceptable throughput was experienced.

PREFACE

This is Volume IV of a four volume report prepared for the Air Force Data Services Center (AFDSC) by the Directorate of Computer Systems Engineering, Deputy for Command and Management Systems, Electronic Systems Division (ESD/MCI). The entire report represents an evaluation and recommendation of the Honeywell Multics system carried out under Air Force Project 6917. Work described in this volume was performed by personnel at ESD/MCI. Special contributions by Mr. Gary Davis, a student at Northeastern University, Boston, Massachusetts, are acknowledged. Computer facilities at the Rome Air Development Center, the MITRE Corporation, the Massachusetts Institute of Technology, and the Air Force Data Services Center were used in the evaluation effort.

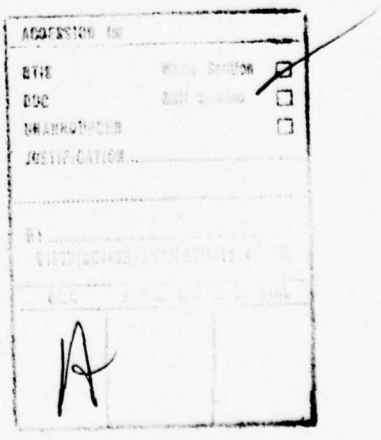


TABLE OF CONTENTS

Section		Page
I	INTRODUCTION	1
II	EVALUATION APPROACH	3
	Introduction	3
	Jobstream Selection	4
	System Gain Factor	4
III	PERFORMANCE TEST	9
	Introduction	9
	IBM 370/155	9
	HIS 635 (GCOS)	11
	HIS 6180 (Native Multics)	13
	HIS 6180 (GCOS Simulator)	15
IV	SUMMARY	19
	Test Results	19
	Evaluation Context	21
	Conclusion	22
Appendix		
A	Gross Diagram of Program Flow	23
B	Description of Program Main Routine	26

LIST OF FIGURES

Figure		Page
1	WWMCCS 14 Benchmark Duplicates Running Simultaneously on 370/155 under OS	10
2	20 WWMCCS 14 Benchmarks Duplicates Started Simultaneously on Dedicated HIS635 at DSC	12
3	20 WWMCCS 14 Benchmarks Duplicates Started Simultaneously on Multics GE645	14
4	20 WWMCCS Benchmarks Duplicates Started Simultaneously on Dedicated HIS 6180 at DSC (Native Multics)	16
5	20 WWMCCS Benchmark Duplicates Started Simultaneously at Dedicated HIS 6180 at DSC (GCOS)	17
6	Comparison Categories	20
7	Transaction Deck	31

SECTION I

INTRODUCTION

The study described herein was motivated primarily by the need of the Air Force Data Services Center (AFDSC) to upgrade the computer security capability available at the Center's computer facility in the Pentagon. AFDSC serves many users of differing interests, whose requirements for classified processing range anywhere from Top Secret down to Unclassified. As discussed in Volume I of this report, (1) the Multics system was the most promising candidate to achieve the desired levels of operational security in a multilevel environment. But at what additional cost in terms of throughput for the jobs submitted?

This question was of particular interest in considering conversion of portions of AFDSC's current workload under GCOS on the HIS 635 to operation under Multics on the HIS 6180. Therefore, ascertainment of the performance impact resulting from this conversion became the primary objective of this study. Another benefit to be derived would be the light to be shed on the proposed changeover of the computer programs possessed by the Office of the Assistant Secretary of Defense for Systems Analysis. OASD (SA) was considering the conversion of these programs from IBM model 360/67 to AFDSC Multics. In addition, since WWMCCS uses GCOS on the HIS 6000 series, this study would provide some insight into WWMCCS workload performance in a Multics environment. Because of this interest, a WWMCCS procurement benchmark job was used for this study.

The goal of this study, therefore, was the acquisition of a realistic feeling for the practical performance aspects that would manifest themselves with use of the Multics system. The approach used for the evaluation was the running of nearly identical jobstreams on the various systems under consideration. This process was used to measure throughput capacity of each in terms of number of jobs per hour for each jobstream. In order to gain a nearly worst-case estimate of the impact on

---

(1) Lipner, Steven B., Multics Security Evaluation: Results and Recommendations, ESD-TR-74-193, Vol I.

Multics performance, a demanding (to Multics) workload was selected. It was demanding in various aspects including: (1) totally batch jobstream on a Multics tuned for interactive processing, (2) compute-bound rather than a reasonable job mix, (3) explicit file operations instead of direct virtual memory file accessing, and (4) use of a modestly emphasized language processor or encapsulation subsystem.

Five distinct test cases were run. In spite of the fact that a highly pessimistic (for Multics) workload had been selected, this study revealed that the Multics system experienced no significant inherent performance degradation.

SECTION II

EVALUATION APPROACH

INTRODUCTION

In addition to security, Multics offers certain advantageous features including effective support for large data bases, real time interactive processing, controlled sharing of information, and characteristics providing a convenient environment for program development. Advances in the state-of-the-art of computer hardware have reduced hardware costs to the point where software spending now frequently constitutes the greater percentage of total acquisition costs. As the ratio of software to hardware costs continues to rise, the foregoing features of Multics become increasingly illustrative of its cost-effectiveness, an advantage, however, that will not be reflected in the throughput measurements performed in this study.

At this point, it should be emphasized that a simple comparison of job throughput capacities does not constitute a cost analysis. Given the different machine environments under which jobs were run, a complete analysis would require obtaining cost figures on the hardware and software involved, a task not attempted here. This report presents a performance evaluation conducted to produce a meaningful comparison of the performance realized from several systems. The results were based primarily on data generated to compute the system gain factor and effective jobs/hr throughput by each system. System gain is detailed and defined in a later section.

Hopefully, an answer might be obtained from a series of experimental runs made on several operating systems with the same jobstream. This jobstream could consist of either one or more existing programs or else some that might be written for the purpose. Such action would, of course, entail the devisal or selection of a method for evaluating the results gained from execution of the programs. Only a relative comparison of system throughputs was desired. The method of assessment needed was not one for yielding results with a high degree of precision. For comparison, not even absolute data was required.

JOBSTREAM SELECTION

The program chosen as a test program was one that has been widely used for test purposes on a number of previous occasions. It had been written initially to serve as one of the benchmark programs for evaluating vendors' equipment during the WWMCCS procurement. For this reason the program was particularly attractive, since it could be expected to be representative of some significant portion of the WWMCCS workload. Designated as No. 14 of the benchmark series, it is the only one among the programs that was written in FORTRAN.

The principal reason for its selection was that it exemplified a nearly worst case that might be processed under operation with the Multics system. As such, it would provide a comparison with current systems that would be indicative of a lower bound on Multics throughput.

Multics FORTRAN was not as highly developed (at the time of this evaluation) as other implementations of FORTRAN available on other systems. Consequently, the selection of a program written in this language introduced an additional qualification of this program for use in illustrating system behavior under the situation of a worst case. Moreover, WWMCCS 14 took advantage of very few of the highly developed capabilities of Multics.

Only this one job type was included in the jobstream, thus facilitating calculation of the jobs per hour capacity of each system. A sufficient number of independent instances of this job were included in the job stream to enable the multiprogramming and multiprocessing capabilities of the system to be effectively used -- to more accurately reflect the true throughput capability. The overall intent was to model a (time) segment within the job stream of each machine.

SYSTEM GAIN FACTOR

The method employed for evaluation of the throughput of each system examined was a measurement algorithm defined by Hoffman and Gwynn (2) to obtain a system gain

---

(2) Hoffman, John M. and Gwynn, Jr., John M. "PRE-SCHEDULING - A Management Tool?", Proceedings of the 1971 Annual Conference, Association for Computing Machinery, Chicago, Ill.

actor applicable to multiprogramming operation. This factor was derived from one somewhat similar, an improvement factor introduced by Hellerman and Smith (3) as a means for determining the response of an operating system to a multiprogrammed workload.

This system gain factor was designed for application in a computing environment consisting (at least in part) of programs run routinely. The elapsed time from start to finish of each of the routine programs is obtained by running each program individually on the machine, that is, without competition with other programs for use of the system's facilities. The sum of these individual times becomes the numerator of a fraction whose denominator is the elapsed time required for a multiprogrammed run of the same group of programs. Thus, system gain is defined as:

$$SG = \frac{\text{Sum of the run-alone elapsed times}}{\text{Multiprogrammed run time}}$$

As Hoffman and Gwynn point out, an SG value larger than unity is indicative of the existence of some advantage in running a given group of programs by multiprogramming, since the group is processed faster this way than the same programs would have been if executed serially. Runs with a different mix of these programs - or with various mixtures with other programs - can yield higher or lower values for the same system.

Obviously, the resulting value depends on the extent to which the programs have either common or uncommon compute-bound and input/output-bound characteristics. Thus, routine programs whose run schedules can be rearranged without serious degradation from the purposes they serve may be grouped for optimum throughput by an empirical process. (In this evaluation, as described below, all the jobs are compute-bound.)

In the analysis presented herein, the WWMCCS 14 program was used for determination of the SG by first timing a run of one copy of the program on a dedicated machine operating under control of a multiprogramming system. Next, a number, n, of copies of the program were run and timed with the machine again in its dedicated

---

(3) Hellerman, H. and Smith, Jr., H. J., "Throughput Analysis of Some Idealized Input, Output, and Compute Overlap Configurations", Computing Surveys, Vol 2, No. 2, June 1970.

state. SG then becomes the quotient of n run-alone times divided by the multiprogrammed run time for n programs. That is:

$$SG = \frac{\text{Run-alone time multiplied by n}}{\text{Multiprogrammed run with n programs}}$$

Note that SG is dimensionless; yet, it furnishes the insight desired for exploratory comparison of systems.

WWMCCS 14 PROGRAM ACTIVITY

Throughout its run time, the major operations performed by the WWMCCS 14 program are mathematical in nature. In response to sets of parameters contained in decks of incoming transaction cards, it selects particular entries from its file of matrices and precedes to process these in a prescribed manner. In so doing, it generates a set of ten eigenvalues and a number of other functions, all of which require considerable computation to produce numerical results.

Though the printout emanating from the processing may appear voluminous (the program outputs a page for each of its transactions), it is not input/output-bound, as might be presumed. Engaged in performing many iterations with the foregoing matrices, this program is in reality highly compute-bound. Consequently, it places a heavy burden on the capabilities of the cpu of the system it is testing. The input/output facilities of the system play a minor role.

A gross flow diagram of the program appears in Appendix A. A detailed description is given in Appendix B.

Some of the details of the WWMCCS 14 program are explained at this point because they are necessary for an understanding of its scope and magnitude, as well as the pertinence to the study of the program attributes described. A few changes to the original WWMCCS Job 14 program were accomplished for the purpose of achieving program status as a purely American National Standard (ANS) FORTRAN program.

The set of ten eigenvalues previously mentioned is computed by an incorporated subroutine eigen, which is called by the main routine. The subroutine is a copy of a portion of IBM's library of scientific subroutines. By

avoiding calls to either the user's own library or to the library of any unique operating system, the execution of WWMCCS 14 is effected purely by sequences of ANS FORTRAN statements making up the job.

After computation of the ten eigenvalues, each value becomes the argument for one of the ten different mathematical functions identified by the program. Each function is then evaluated by the program. Evaluation of five of these is accomplished by invocation of ANS FORTRAN functions. Two others - after some slight manipulation - invoke exponential or natural logarithmic routines, also ANS FORTRAN functions.

The remaining three mathematical functions depend for their evaluation on calls to subroutines that have been embedded in the WWMCCS 14 program. The program instructions that make up these subroutines were taken in whole or in part from the same scientific package that contained the eigen routine. Here again, the programs appear to the computer as ANS FORTRAN.

Further detail relative to the above can be found in the appendices.

Another step taken in the modification of the program statements in order to qualify them as standard FORTRAN instructions was to remove the special IBM direct access statements present in the program. These were the READ and WRITE statements in which the initial parameter is an integer constant followed immediately by an apostrophe. The apostrophe indicates that the integer constant is a data set reference number identifying a file previously established by a DEFINE FILE statement.

This statement, which defines the characteristics of the data set that is to be read into internal storage or written from it, had been used to form the Compute File, the principle file toward which the program's activities are directed. It consists of one hundred ten-by-ten matrices, all of whose elements are signed, two-digit integers. Matrices of this file are extracted, processed, and returned or replaced by the program.

To render the file accessible by standard FORTRAN statements, it was necessary to position the entries of the file where they could be accessed by standard READ/WRITE statements. This necessitated placing the file in core memory (or virtual memory for systems with demand paging).

This action did not in any way change the identity of the elements or the computational operations performed on them. It did, however, require minor rearrangements in the indexing of the variables controlling the loops used for reading and writing the elements.

A final step taken to assure that all program statements were in the ANS FORTRAN language was to remove the END=d, which appears as a parameter in the READ instructions for five of the six cards in the input transaction deck. The purpose of this parameter is to transfer control to the statement designated by d in event the end of the data set is encountered.

In the WWMCCS 14 program, as formulated, such an encounter occurs only when there are no more transactions to be processed. Punching a dummy number in the last card of the final transaction and adding a conditional transfer instruction to STOP achieved the same objective.

SECTION III

PERFORMANCE TESTS

INTRODUCTION

The tests, whose descriptions follow, generated data for computation of the system gain factor and effective jobs/hour. It is important to note that no restrictions were imposed on the systems tested. Each system was presented with all copies of the job simultaneously. Subsequent processing of these jobs was controlled by the system. Thus any queuing of input or scheduling of resource allocation was controlled by each system itself.

IBM 370/155

The first test was made on an IBM 370/155 at the computer facility of the MITRE Corporation in Bedford, MA. A special run with ten copies of WWMCCS 14 was performed while no other programs were present on the machine. Moreover, HASP, TCAM, and TSO had been rendered temporarily inoperative to remove their resource demands. Results are shown in Figure 1.

Prior to starting the run, the card deck for the entire program had been read in. Each of the ten card images created from it had been assigned a different job name. After nearly simultaneous initiation, the jobs competed with one another throughout their runs. Later, after the test had been completed, the system writer produced printouts of the results.

Nine of the jobs started within a few seconds of one another, but the tenth did not begin until 4.3 minutes after the beginning of the first job. Though it finished last, the aggregate time for compilation, link-editing, and execution constituted nearly the shortest lapse of time from start to finish, since this job did not have to compete as strongly for the resources of the system.

Indeed, its execution step consumed only 36 seconds, or one second in excess of the time needed for execution of the object program within the job when the job was processed by itself on the dedicated system. (See lower left of Figure 1.)

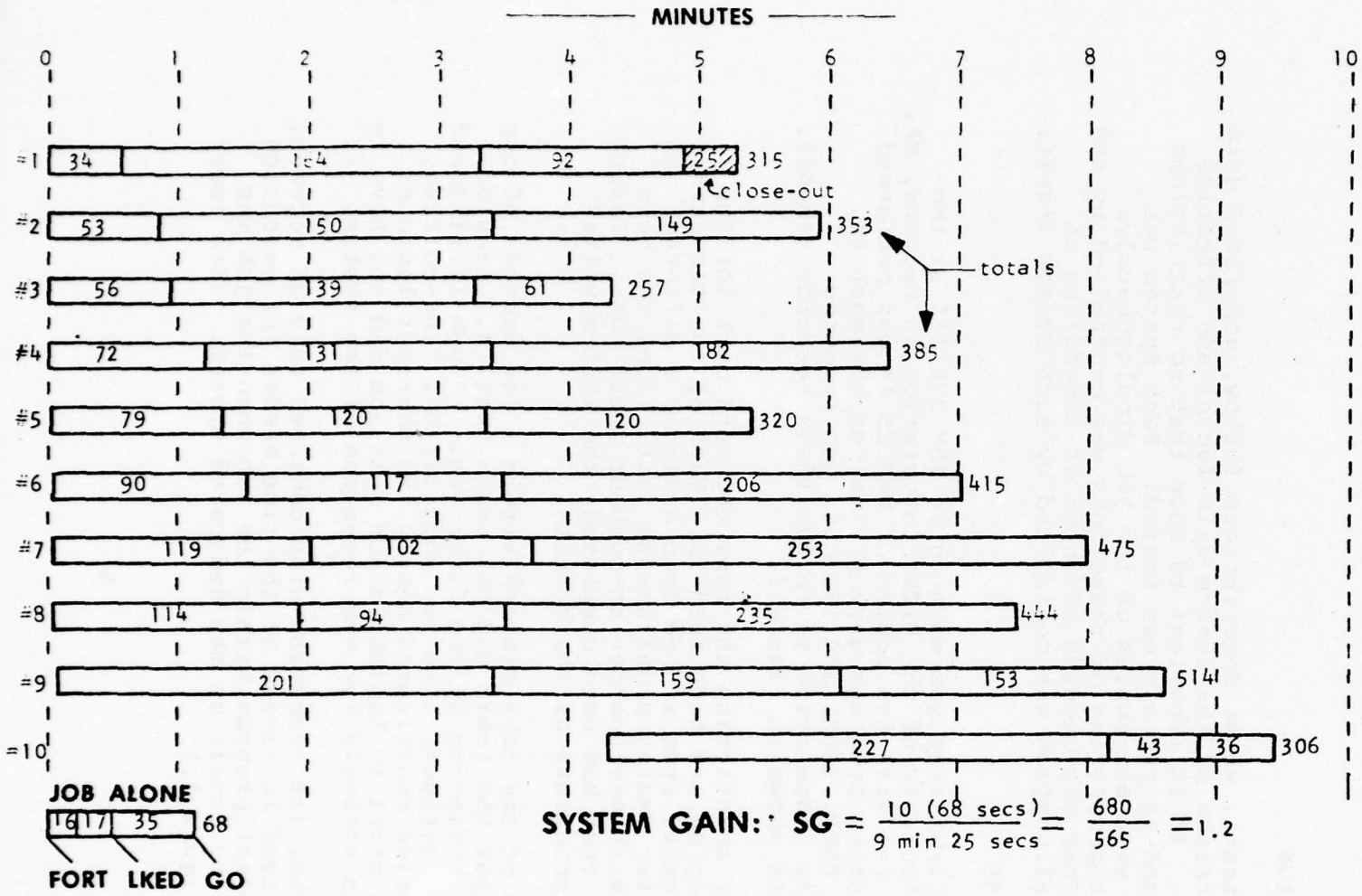


FIGURE 1

WWMCCS14 BENCHMARK DUPLICATES RUNNING SIMULTANEOUSLY ON 370/155 UNDER OS

(FIGURES ON BARS ARE SECS. OF ELAPSED TIME)

When one job was run alone all resources of the system were immediately available. There was no waiting time between steps. The seconds of time marked on the bar for each job represent the elapsed time from the start to the finish of each step. The intervals between steps were too short for recording in the diagram.

The last job to finish was the tenth at a time 9.4 minutes after the first job began. Run alone, the job took 68 seconds as shown in Figure 1. System gain equaled 1.2.

HIS 635 (GCOS)

The second test was performed on the Honeywell 635 at AFDSC. Since this is a dual processor, the number of jobs was doubled to avoid as much loss of accuracy as possible. Again, WWMCCS 14 was executed with the system dedicated to this job alone. Control was exercised from an operator console physically situated at AFDSC. Results are shown in Figure 2.

Prior to beginning this experiment, twenty copies of the job had been entered into the HIS 635, and each copy had been assigned a different job identification. Again, after near simultaniety of initiation, the jobs ran on the dedicated machine and competed with one another for the system's resources.

The output listings of these runs provided only two groups of timings: the beginning and end of the compilation and the beginning and end of the combined time to load (link-edit) and execute. Though the time lapses between the two groups were distinguishable, they were very small in comparison to the overall times for the runs themselves. Consequently, they could not be depicted within the time scale selected for presentation of results in Figure 2.

Observe that the first six jobs started at nearly the same time and that the initiations of the following five occurred at times fairly close together. The next three began at times even closer to one another. The fifteenth job did not start until the eighth had completed. It was followed in close order by the remaining five runs.

The last job to finish was the twentieth at a time 13 minutes and 44 seconds after the first job began. Run alone, the job took 86 seconds as shown in Figure 2.

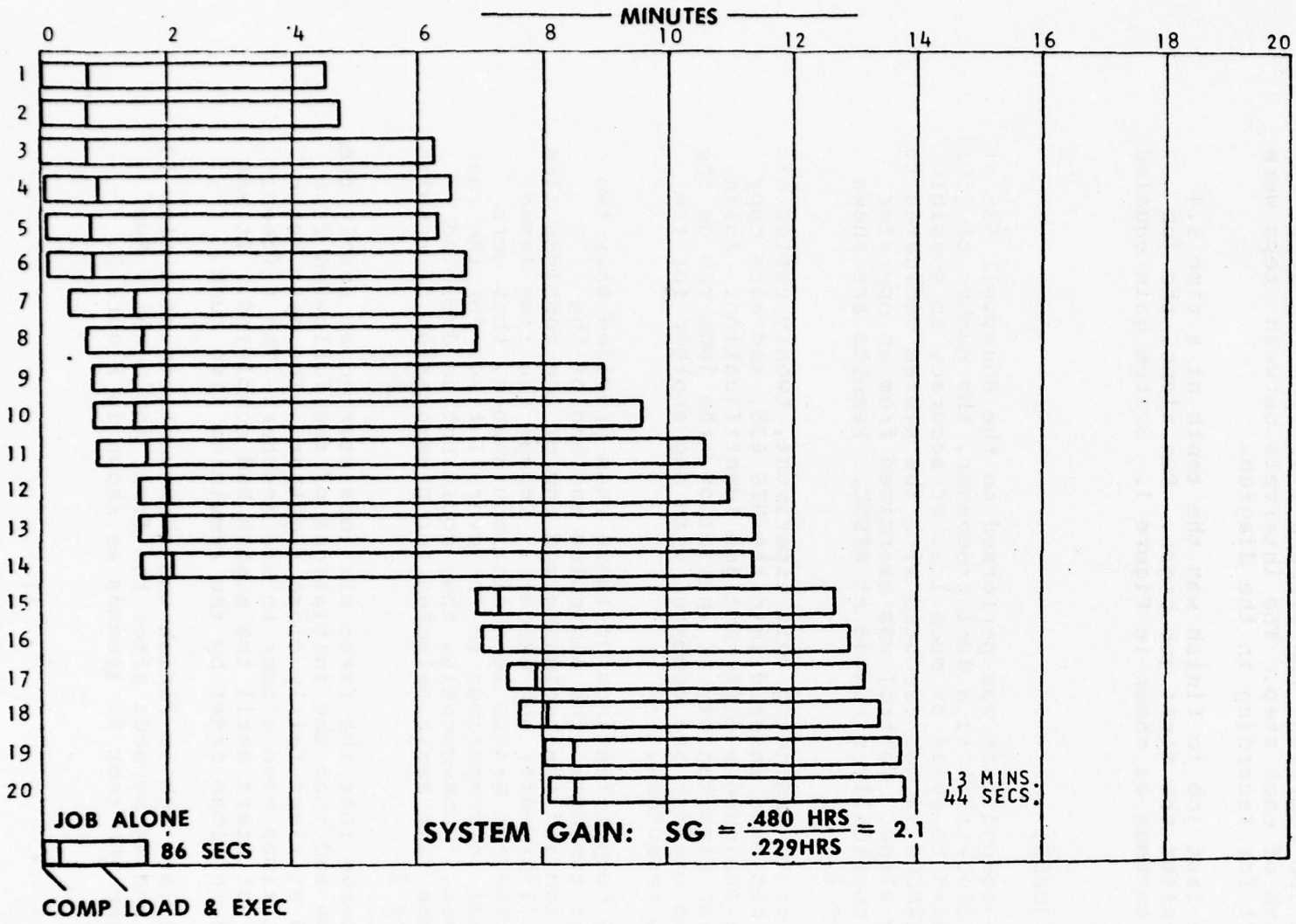


FIGURE 2

20 WWMCCS14 BENCHMARK DUPLICATES STARTED SIMULTANEOUSLY ON DEDICATED HIS635 AT DSC

System gain equaled 2.1.

HIS 645 (Native Multics)

The third test was performed on the Multics system at MIT before the Honeywell 645 was replaced by the Honeywell 6180. Control was exercised from a terminal physically situated on the same premises in order to minimize the time the machine would have to be kept in a dedicated state. Communication by telephone from a terminal remotely located would have consumed extra time in carrying out the experiment. Results are shown in Figure 3.

Twenty copies of WWMCCS 14 were invoked from the Multics file system, where the job had been previously stored. Each copy was given an individual jobname. All jobs had started before the first, or earliest, had finished its compilation. In fact, the fourteenth, which started two minutes after the first, completed its compilation well in advance of any of the others.

Binding, which is tantamount to link editing, required less time than did either the compile or execute steps. All jobs (except the sixth, which Multics scratched due to a HIS 645 hardware error) started and stopped within a few minutes of one another.

The last job to finish was the eleventh at a time, 42 minutes and 32 seconds after the first job began. Run alone, the job took 217 seconds as shown in Figure 3. System gain equaled 1.6.

HIS 6180 (Native Multics)

The fourth test was run at AFDSC on the Honeywell 6180, the processor currently used by AFDSC for the implementation (4) of Multics. Control was exercised from

(4) Honeywell's present practice is to implement Multics systems on two of their more advanced processors, the 68/80 and 68/60. These units are somewhat faster than the 6100 and contain cache memory, a feature proven useful but not essential for implementing security controls. The processor at AFDSC, although designated a 6180, has been modified to assimilate some of the advanced features of these processors including cache memory.

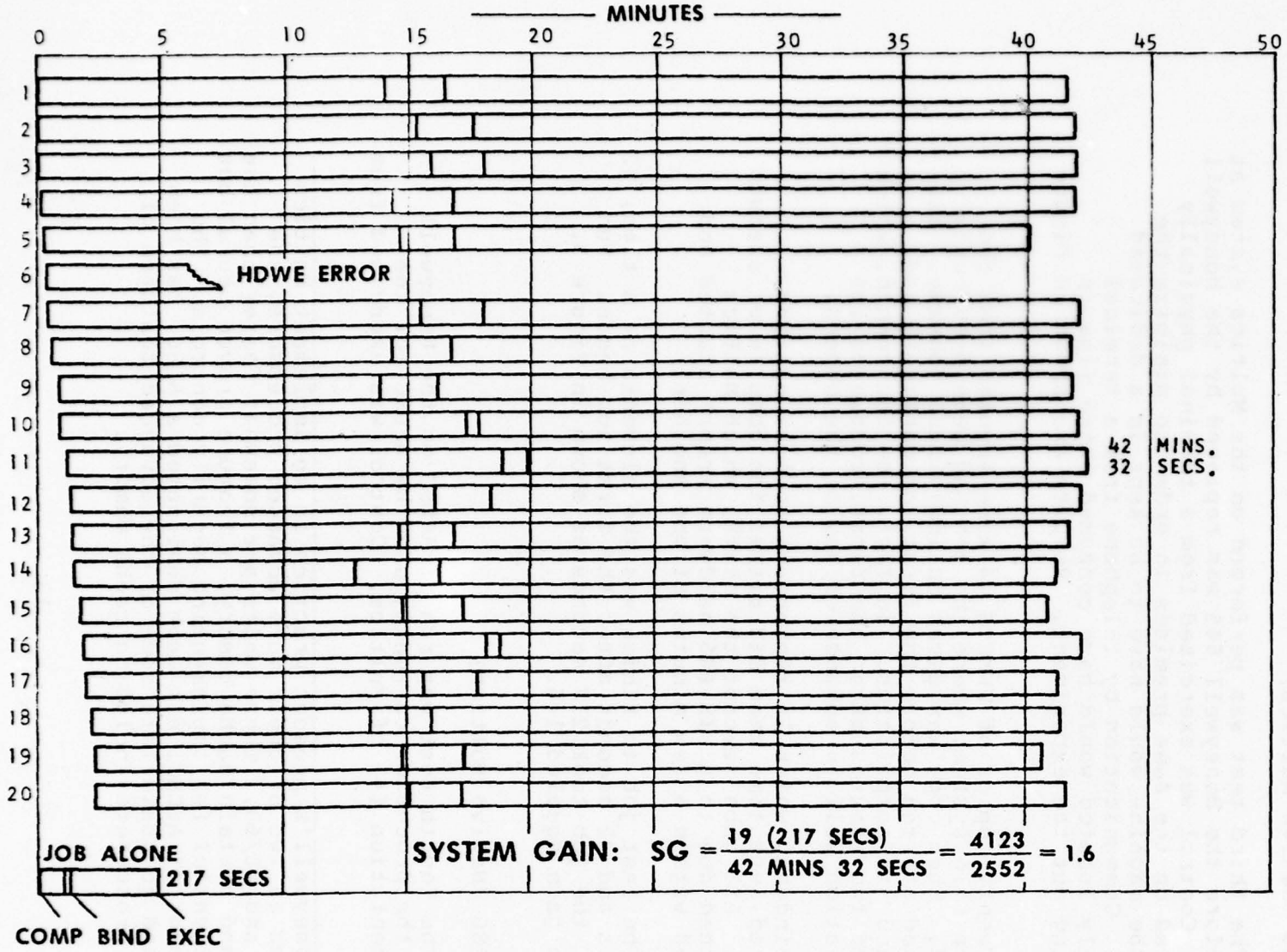


FIGURE 3

20 WWMCCS14 BENCHMARK DUPLICATES STARTED SIMULTANEOUSLY ON MULTICS GE645 (ONE FAILURE)

a remote terminal located at ESD. The output listings were programmed to provide three groups of timings: compilation, loading, and execution. Again, the system was dedicated.

As before, twenty copies of the job, each individually labelled, were invoked from the Multics file system. Competition began for use of the system's resources. Pesults are shown in Figure 4.

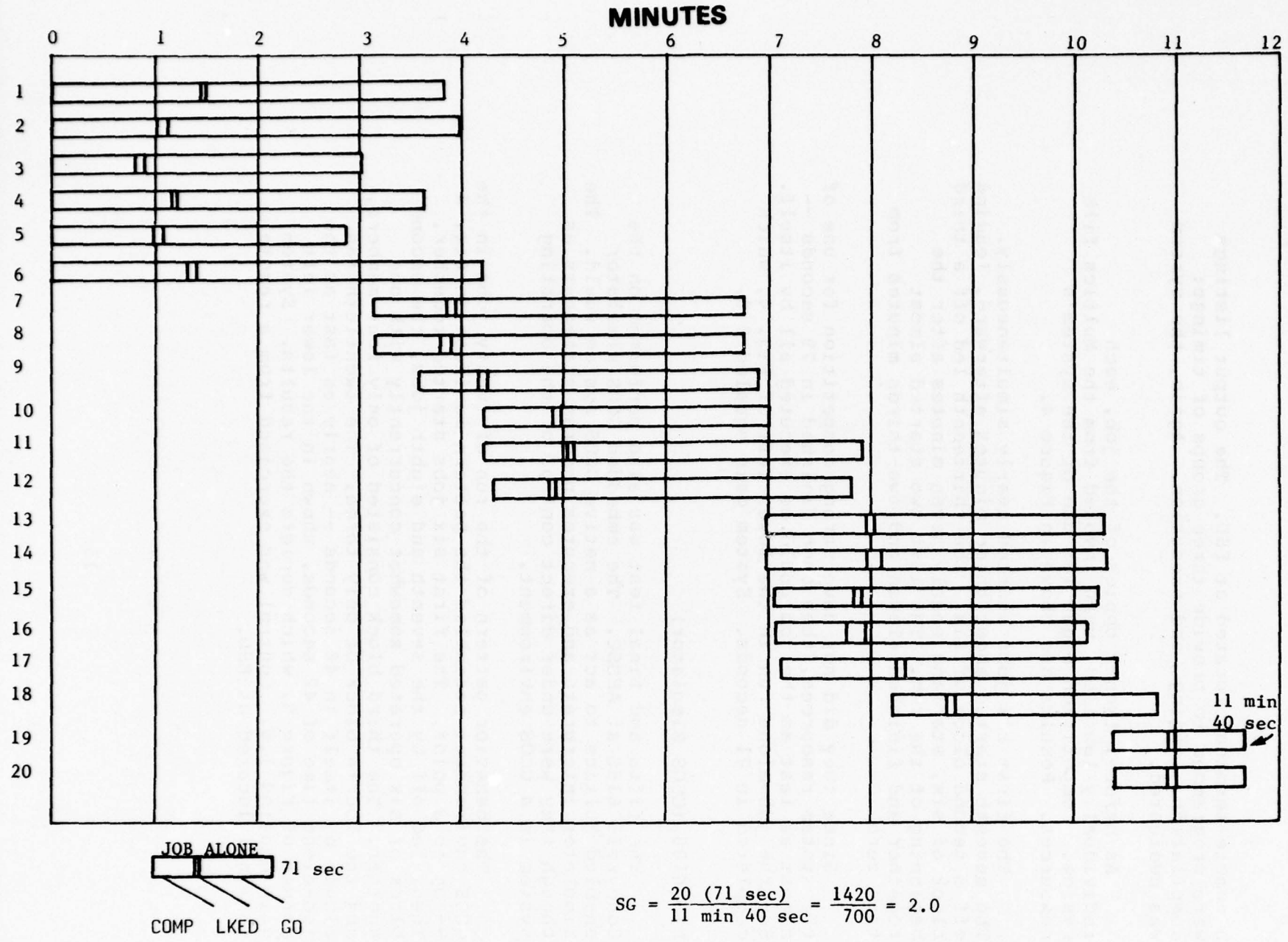
The first six jobs started nearly simultaneously. The seventh started about three minutes afterward, leading off a second block of six. The thirteenth led off a third block of six, starting nearly seven minutes after the beginning of the run. The last two started almost together and finished eleven and two-thirds minutes from time zero.

Since they did not have strong competition for use of the system resources, they each finished in 79 seconds -- almost as fast as the job could be executed all by itself. See the job-alone run in the lower left of Fig. 4, which completed in 71 seconds. System gain equaled 2.0.

HIS 6180 (GCOS Simulator)

The fifth and final test was also performed on the Honeywell 6180 at AFDSC. The embedded GCOS simulator enabled Multics to act as a native GCOS system would. The simulator interprets and executes program statements as though they were under direct control of the operating system in a GCOS environment.

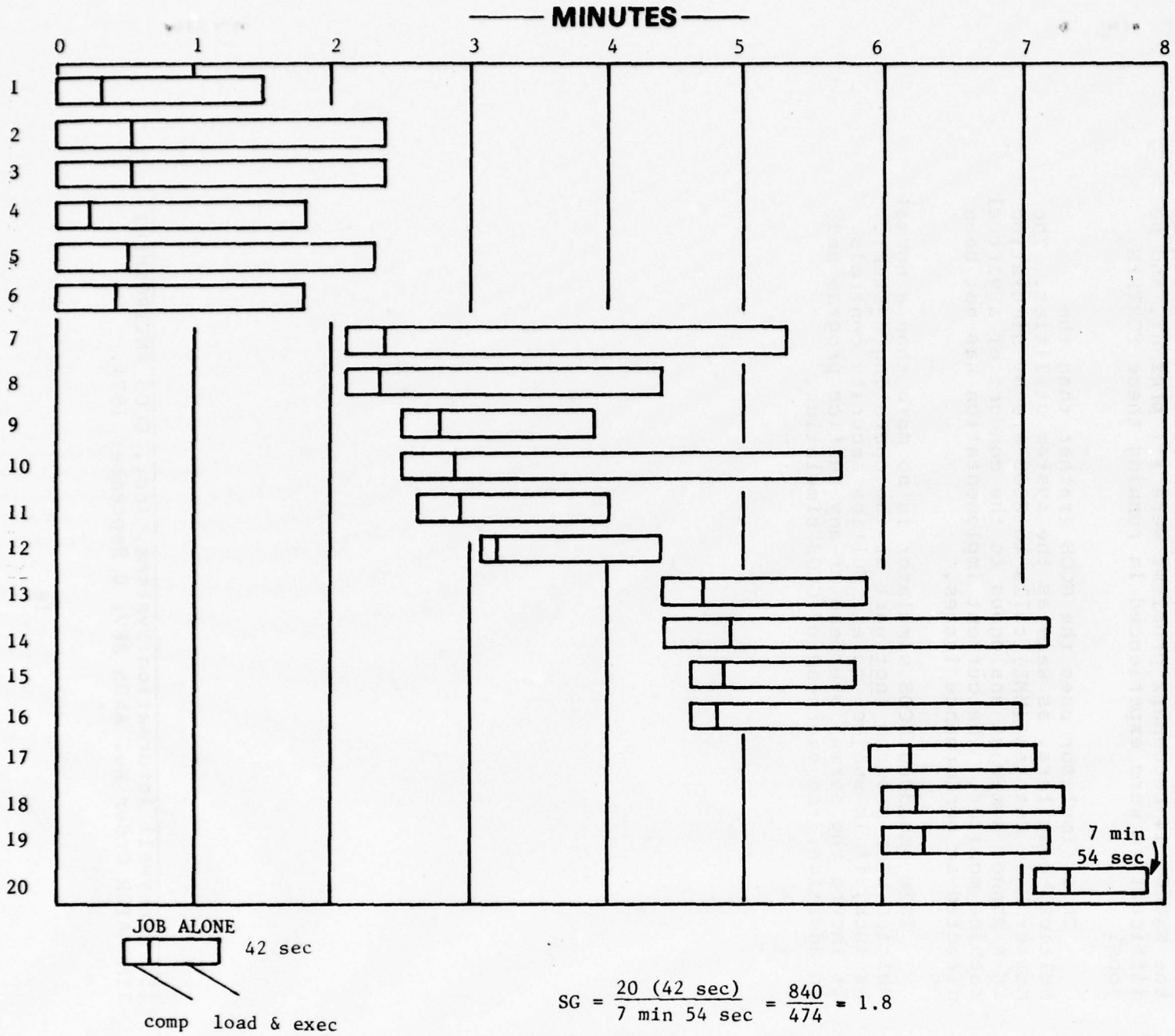
The behavior pattern of the run of twenty jobs in the GCOS environment resembled the pattern manifest in Fig. 4 -- up to a point. The first six jobs started together. Then, led off by the seventh and eighth jobs, the second block of six operated somewhat concurrently with one another. The third block consisted of only four members, and the fourth block of only three. The twentieth ran almost by itself in 48 seconds -- nearly as fast as the job-alone time of 42 seconds, shown in the lower left corner of Figure 5, which depicts the results. System gain equaled 1.8. Control was exercised from a remote terminal located at ESD.



$$SG = \frac{20\ (71\ \text{sec})}{11\ \text{min}\ 40\ \text{sec}} = \frac{1420}{700} = 2.0$$

FIGURE 4

20 WWMCCS BENCHMARK DUPLICATES STARTED SIMULTANEOUSLY ON DEDICATED HIS6180 AT DSC (NATIVE MULTICS)



$$SG = \frac{20\ (42\ \text{sec})}{7\ \text{min}\ 54\ \text{sec}} = \frac{840}{474} = 1.8$$

FIGURE 5

20 WWMCCS14 BENCHMARK DUPLICATES STARTED SIMULTANEOUSLY ON DEDICATED HIS6180 AT DSC (GCOS)

The GCOS simulator (5) was constructed by Honeywell primarily for internal use. It provides a GCOS compatible environment in Multics. Not all of the GCOS capabilities are supported. However, those necessary for execution of the WWMCCS 14 benchmark problems were all present, and no difficulties were experienced in running these FORTRAN jobs.

This simulator uses the GCOS (rather than the Multics) compilers, as well as the system utilities. The Master Mode Entries (MME) calls to GCOS are by simulation in a manner somewhat analogous to the conduct of a virtual machine monitor. The current implementation has not been directed at performance issues.

This embedded GCOS simulator is no more than a normal Multics user program (not part of the operating system). As such, it is subject to all Multics security controls. It incurs the normal overhead of any Multics program and, in addition, the overhead of GCOS simulation.

(5) Honeywell Information Systems, Inc., GCOS ENVIRONMENT SIMULATOR Order No. AN05 REV. 0 September 1974.

SECTION IV

SUMMARY

TEST RESULTS

Results of the performance tests described in Section III are tabulated below. To aid in making comparisons, the jobs completed per total time for each run have been reduced to a common denominator: effective jobs per hour.

PERFORMANCE TEST RESULTS

Date	Facility	System Gain	Jobs Per Time	Effective Jobs/Hour	Software Type
Feb 73	IBM 370/155 at MITRE	1.2	10/565 seconds	63.7	Non-Multics Non-GCOS
May 73	HIS 635 at DSC	2.1	20/824 seconds	87.4	GCOS
Jul 73	HIS 645 at MIT	1.6	19/2552 seconds	16.8	Native Multics
Apr 76	HIS 6180 at DSC	2.0	20/700 seconds	102.9	Native Multics
Apr 76	HIS 6180 at DSC	1.8	20/474 seconds	152.0	Embedded GCOS on Multics

Comparisons among pairs of computer facilities having different software implementations are described below for each of three categories. These categories are identified by the numbered connecting paths in Figure 6. They correspond with the descriptions below.

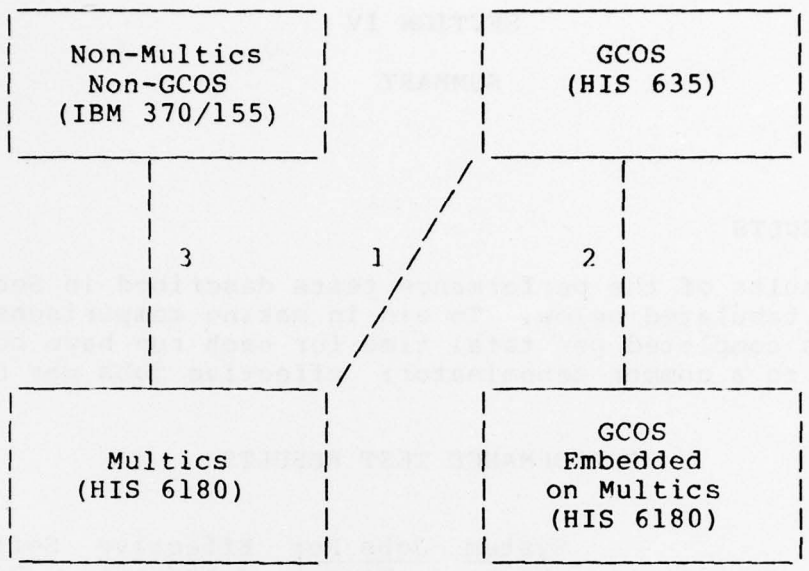


Figure 6
COMPARISON CATEGORIES

Category 1) The performance of native Multics (April 1976), as currently implemented on the HIS 6180 at AFDSC, yielded a System Gain factor comparable to that resulting from the run with GCOS at AFDSC (May 1973). This result contrasted with that produced by another run (July 1973) with MIT's Multics implemented at that time on an HIS 645. Examination of the results of the test run show that Multics was then a system that was considerably slower.

Changeover to the HIS 6180 substantially improved the software and enabled Multics to perform on a level comparable to that of GCOS. The result signifies the effect of major improvements in Multics performance. Before evaluating Multics software, the version (earlier or later) should be ascertained.

Category 2) The performance of the Multics embedded GCOS run on an HIS 6180 exhibited marked improvement when compared to the GCOS run on the older HIS 635. Such was to be expected due to the 6180's more advanced hardware, which invokes the normal underlying Multics mechanisms. Even with the overhead of simulating GCOS, performance was still better than in native Multics. This result indicates that the Multics FORTRAN compiler (used in the

native Multics on the HIS 635) did not perform particularly well with regard to either compilation or execution of the generated object code.

Category 3) The earliest test (Feb 1973) was performed on MITRE's IBM 370, model 155, which is not a dual processor. Since its software system utilizes neither Multics nor GCOS, it served to typify general systems software outside the Multics/GCOS environment. Matched with this run as a reference, the recent run (April 1976) under native Multics on the HIS 6180 at DSC produced significantly favorable results, as shown in Table 1.

EVALUATION CONTEXT

The study was motivated by the realization that an upgrading of the security status of various systems in use at AFDSC had become necessary for the effective operation of those systems, both present and future. The upgrading, aside from the inherent operational capabilities to be realized from such a conversion to Multics, would enable the reduction of several major expenses that are now required to maintain security. The two most visible inefficiencies involve the redundancy of equipment and consequent extra downtime.

At present, a facility desiring the ability to securely process information at more than one security level usually operates in at least one of the three environments that follow:

1. System high clearances (all users of the system must be cleared to the highest security level being processed)

2. Duplicate equipment (a separate processing system is required for each security level processed)

3. Periodic color changes (all evidence of one security level must be purged from the system when preparing for a different level)

The computer facility at AFDSC uses all three of these environments.

Implementation of effective multilevel security controls would significantly reduce the need for system high clearances and duplicate equipment. Downtime due to color changes would be eliminated.

Performance of a cost analysis was not the purpose of this report. Examples of cost-effectiveness were discussed only for the purpose of showing the impact a Multics environment would have upon AFDSC. The main concern of this report was to determine the effect on throughput to be expected if conversion to a Multics environment should be accomplished. A worst case analysis was undertaken in order to indicate the maximum effect any possible degradation would have on the system.

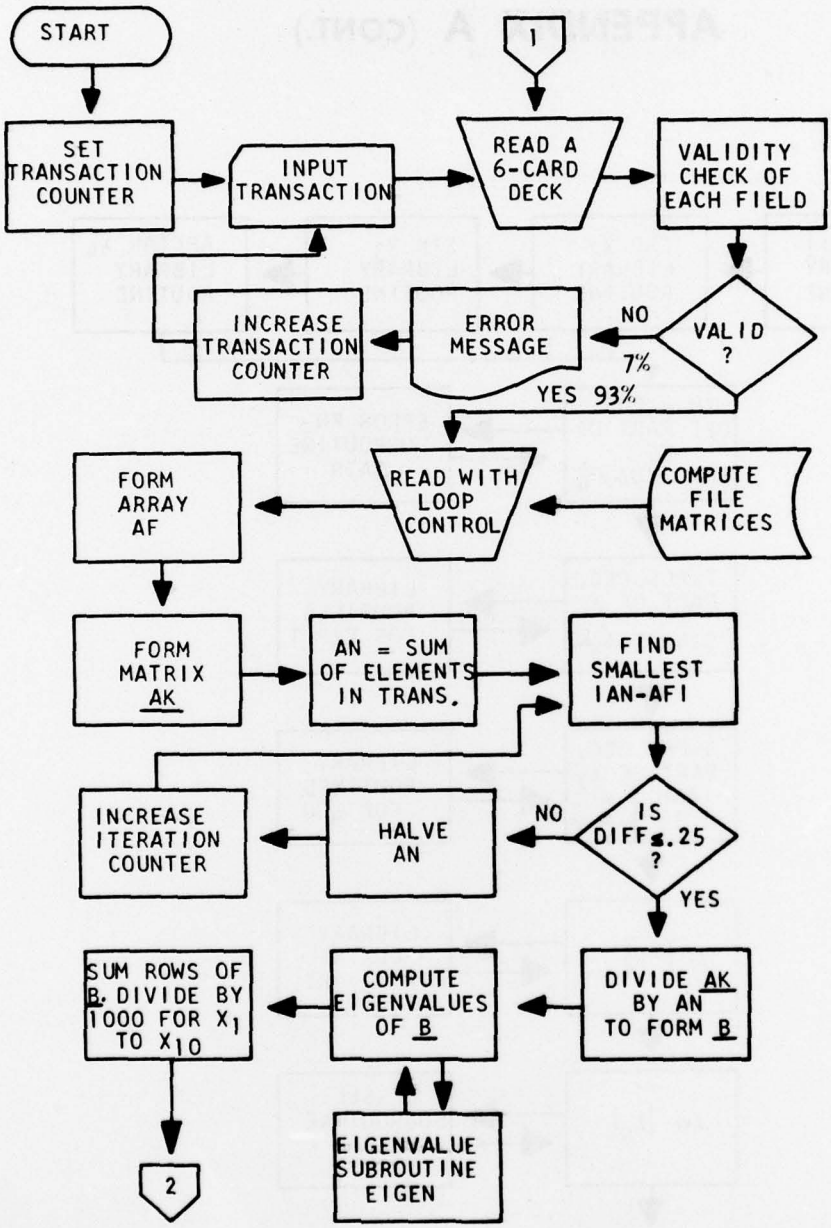
CONCLUSION

The premise presented at the beginning of this study was that AFDSC needed to acquire a multilevel processing capability in order to better meet its functional objectives. Such a security capability conceivably could hamper the performance of a system so equipped. The question to be answered was the following: Could AFDSC meet its processing needs based on Multics without unacceptable degradation of performance?

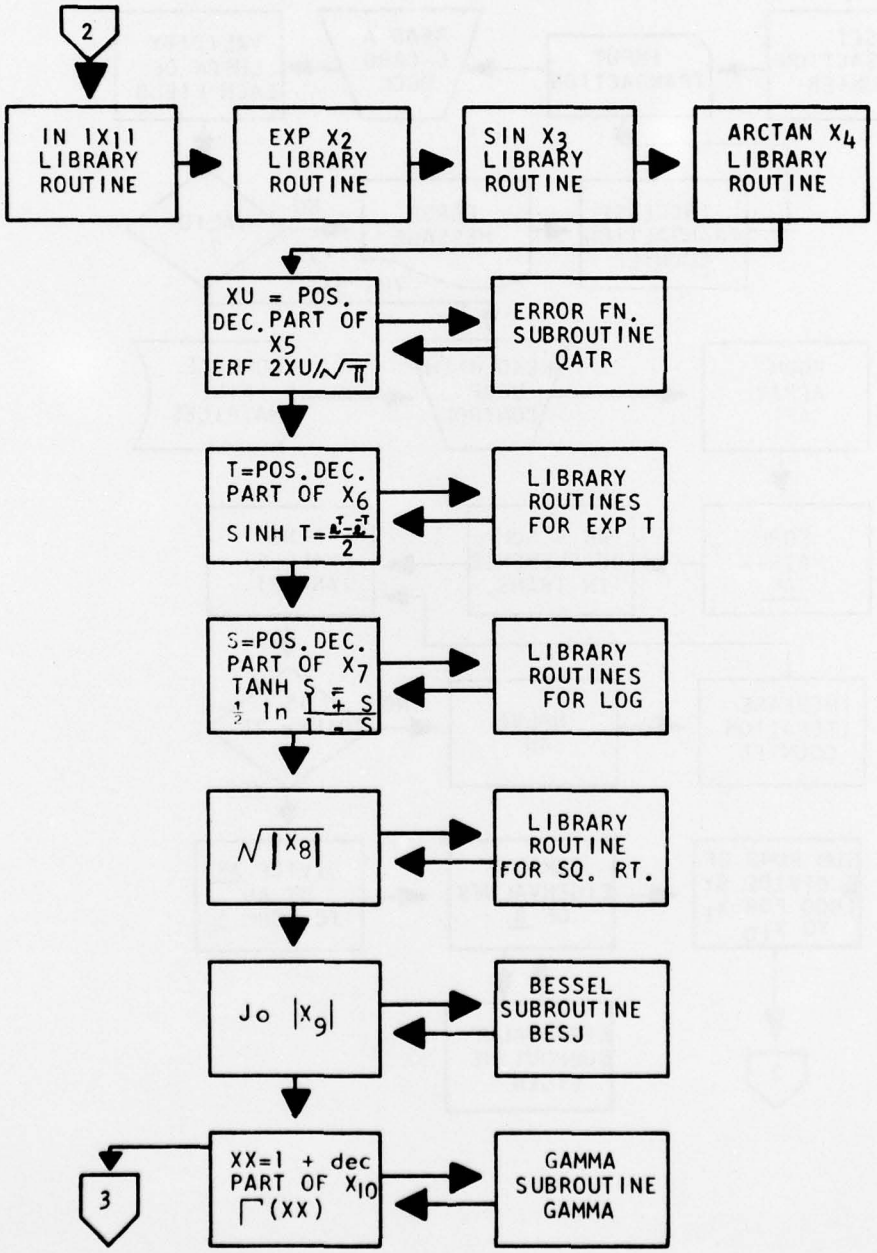
The test results indicated that even in the unrealistic, worst-case conditions introduced in this evaluation, no unacceptable degradation in throughput would be experienced. The advantageous features of Multics including real time interactive processing, controlled sharing of information, and an effective capability for processing large data bases would all be available without having to accept a degraded level of performance in a tradeoff. Overall, in a convenient environment for program development, Multics should prove cost-effective for both present and future system development and production while at the same time meeting the necessary security requirements.

APPENDIX A

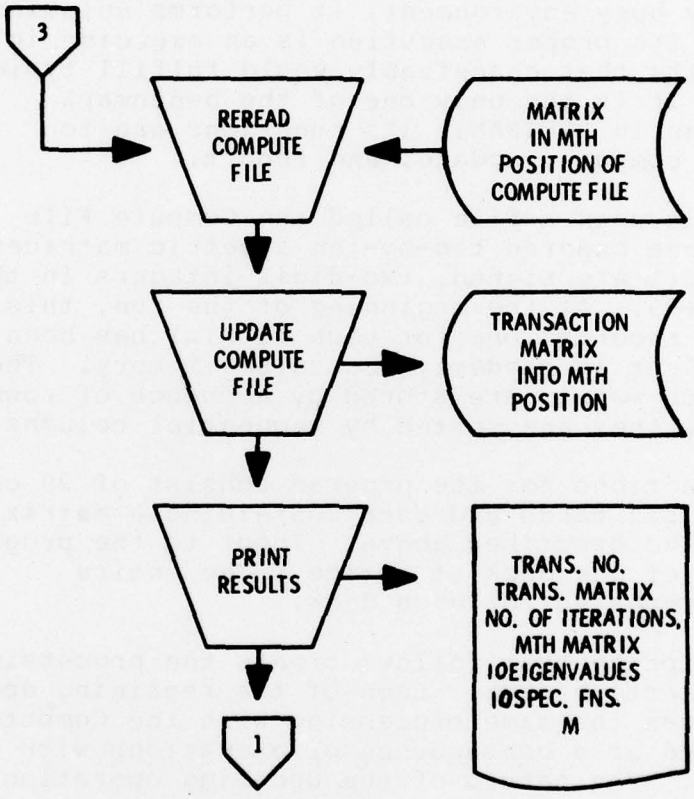
GROSS DIAGRAM OF PROGRAM FLOW



APPENDIX A (CONT.)



APPENDIX A (CONT.)



APPENDIX B

DESCRIPTION OF PROGRAM

MAIN ROUTINE

The program for WWMCCS 14 is exemplary of the class of programs found in scientific systems whose primary purpose is extensive mathematical computation. Designed to demonstrate the type of activity that routinely takes place in a very busy environment, it performs voluminous calculations. Its proper execution is an exercise in producing results that conceivably would fulfill typical requirements. It is the only one of the benchmark programs written in FORTRAN. Its functions are to extract, edit, compute, update, and report.

For data it uses a file called the Compute File consisting of one hundred ten-by-ten symetric matrices all of whose elements are signed, two-digit integers in the range, -99 to +99. At the beginning of the run, this file of one hundred records (one for each matrix) has been read in and is resident in randomly accessible memory. The elements in each matrix are stored by sequence of rows. Within each row they are sorted by sequential columns.

The transactions for the program consist of 29 card decks, each of six cards and each containing a matrix of the size and kind described above. Input to the program is at the rate of one deck at a time. The entire processing is repeated for each deck.

The description that follows treats the processing of the first transaction deck. Each of the remaining decks in turn undergoes the same processing with the Compute File now updated as a consequence of operations with previous decks. The nature of the updating operation will be described later.

As each card of the six cards comprising a transaction deck is read in, all fields are tested for proper content. Erroneous input is signalled by generation of an appropriate error message and the transaction is then skipped. Two of the decks are purposely incomplete and give rise to an error report.

Next, the elements in each of the one hundred matrices within the Compute File are totalled. Each sum is placed in one of the one hundred members of the array

AF.

A new matrix AK is now formed. Each element (i,j) within it becomes the total of the one hundred elements that occupy the same (i,j)th position within each matrix of the Compute File, that is, the position designated by the same row and column.

At this point the total of the elements in the current transaction matrix is obtained. This total is then designated AN and retained for later use.

From this total, each of the sums in the array AF produced from the matrices is subtracted in turn to yield one hundred differences. As each difference is obtained, its absolute value is compared with the smallest absolute value among the differences previously obtained. The ordinal number of the matrix whose element sum gave rise to the minimum value is temporarily retained in M for a later printout, to take place after the test described in the next paragraph has been satisfactorily completed.

The minimum attained is compared with the quantity, 0.25. If larger than this quantity, the total AN obtained by summing the elements in the transaction matrix is halved. Every step in the preceding paragraph is then repeated using this halved value. Previous results are replaced. For later printout, a count is kept of the number of iterations.

The data provided is such that repeated interations of this process will eventually yield a value less than or equal to 0.25. A count is kept of the number of interations required to achieve this result. It is printed as part of the output report following completion of processing of the current transaction matrix.

Next, each element in the matrix AK is divided by the quantity AN, as diminished by repeated halving. Thus, a new ten-by-ten matrix called B is generated for use in further processing.

A breakpoint in the procedure now takes place during which the main program calls a subroutine to compute the ten eigenvalues of B. These are returned by the subroutine and held for report at the completion of operations with the current transaction matrix.

On return to the main program the elements in each row of B are summed and each of the ten sums is then

divided by 1000. The resulting ten values, X(I), are used in sequential order to form arguments for the evaluation of the special functions as tabulated below. Unless otherwise mentioned, system library routines are employed for these calculations.

ARGUMENT	FUNCTION
X(1)	Natural logarithm of magnitude.
X(2)	Exponential
X(3)	Sine.
X(4)	Arctangent.
Fractional Part of X(5)	Error function of magnitude (by subroutine).
Fractional Part of X(6)	Hyperbolic sine of magnitude.
Fractional Part of X(7)	Hyperbolic anti-tangent of magnitude.
X(8)	Square root of magnitude.
X(9)	Bessel function of order zero (by subroutine)
Fractional Part of X(10) +1	Gamma function of magnitude (by subroutine)

This is the end of the computational work performed by the program. The matrix finally flagged with the value M is now read into core and held for output later on. Its position in the Compute File then becomes occupied by the current transaction matrix, thereby updating the file.

The printed report begins on a new page headed by the transaction deck number and followed by a printout of the transaction matrix itself. Next is the count of iterations previously cited. Following this is the replaced matrix, the ten eigenvalues, and a column of the ten special function values. The last figure is the value M, the ordinal number of the position in the Compute File updated by replacement of the matrix occupying this position.

Processing now continues by reading in the next transaction deck and repeating the foregoing sequence of operations, this time with the updated Compute File. It ceases after the last transaction has been processed.

SUBROUTINES

The four subroutines explained below are copies -- sometimes modified versions -- of those of similar name appearing in the manual entitled Scientific Subroutines published by IBM Corporation. The versions herein are all FORTRAN routines, incorporated as part of the WWMCCS 14 program and invoked as ordinary subroutines. They are therefore nowise dependent on any particular operating system.

When the subroutine EIGEN is called, the hundred elements in the matrix $\underline{B}$ are passed to it. It then computes the eigenvalues by the Jacobi Method. The unneeded portion, which calculates the eigenvectors, has not been included in the subroutine used in the benchmark program. The method used is applicable only in event the matrix is symmetric and contains no complex values among its elements, as in this case.

The input matrix, $\underline{B}$ is automatically accepted by this routine and the ten eigenvalues are automatically returned in an array AA for eventual printout as part of the report for the current transaction.

Evaluation of the integral within the error function is performed by QATR, a utility routine for numerical quadrature, which uses the trapezoidal rule. A few instructions have been added at the beginning of this subroutine for the purpose of identifying the function that is to be numerically integrated. In this case the function is the exponential of the negative value of the square of the dummy variable. The bound on the absolute error is also specified.

When the integration has been completed, the result is returned to the main program. Here it is immediately multiplied by two and divided by the square root of pi to satisfy the definition of the error function. The resulting value is stored in FX (5) for later printout as part of the report for the current transaction.

Computation of the Bessel function, Jn (x), where n is the order and x is the argument, is accomplished by the subroutine called BESJ. Its code is founded on an algorithm reported by Stegun and Abramowitz in a 1957 mathematical journal, which in turn stems from an earlier method published by Miller in the introduction of a British table of Bessel functions.

The input to the GAMMA subroutine is the fractional part of the tenth value of X plus one. This subroutine receives its absolute value as an argument.

Computation is by evaluation of a seventh degree polynomical whose coefficients have been expressed to seven significant digits. The correct value for the coefficient of the second degree term is 0.9858540. However, the value used in the subroutine was copied incorrectly as 0.985840. Inasmuch as the arguments were small, it is not expected that the results (obtained to five places) were affected in any position except the last - if at all.

TRANSACTION FORMAT

A typical input transaction record is illustrated in Figure 7. It is a six-card deck containing 100 signed integers in the range -99 to +99, similar to the elements in each matrix of the Compute File.

In the first card, R in column 1 indicates that the transaction comes from a card reader. Columns 2 and 3 indicate that the record is to be used with Job 14. Columns 4 and 5 indicate that File 13 will be affected by the transaction. Z in column 6 indicates that the input is computational.

Columns 76-80 in each card except the last, contain the numbers of the transaction and of the card within it. The sixth card contains the word, END in its last three columns.

The deck is read in by six sequential read instructions. The final three columns of each of the cards is checked for appearance of the word END. If this is found prematurely, the program branches to an error message to the effect that the data set is incomplete. In this event, the sixth read instruction is not reached. The program increases the ordinal number of the transaction and then returns to the beginning to read a

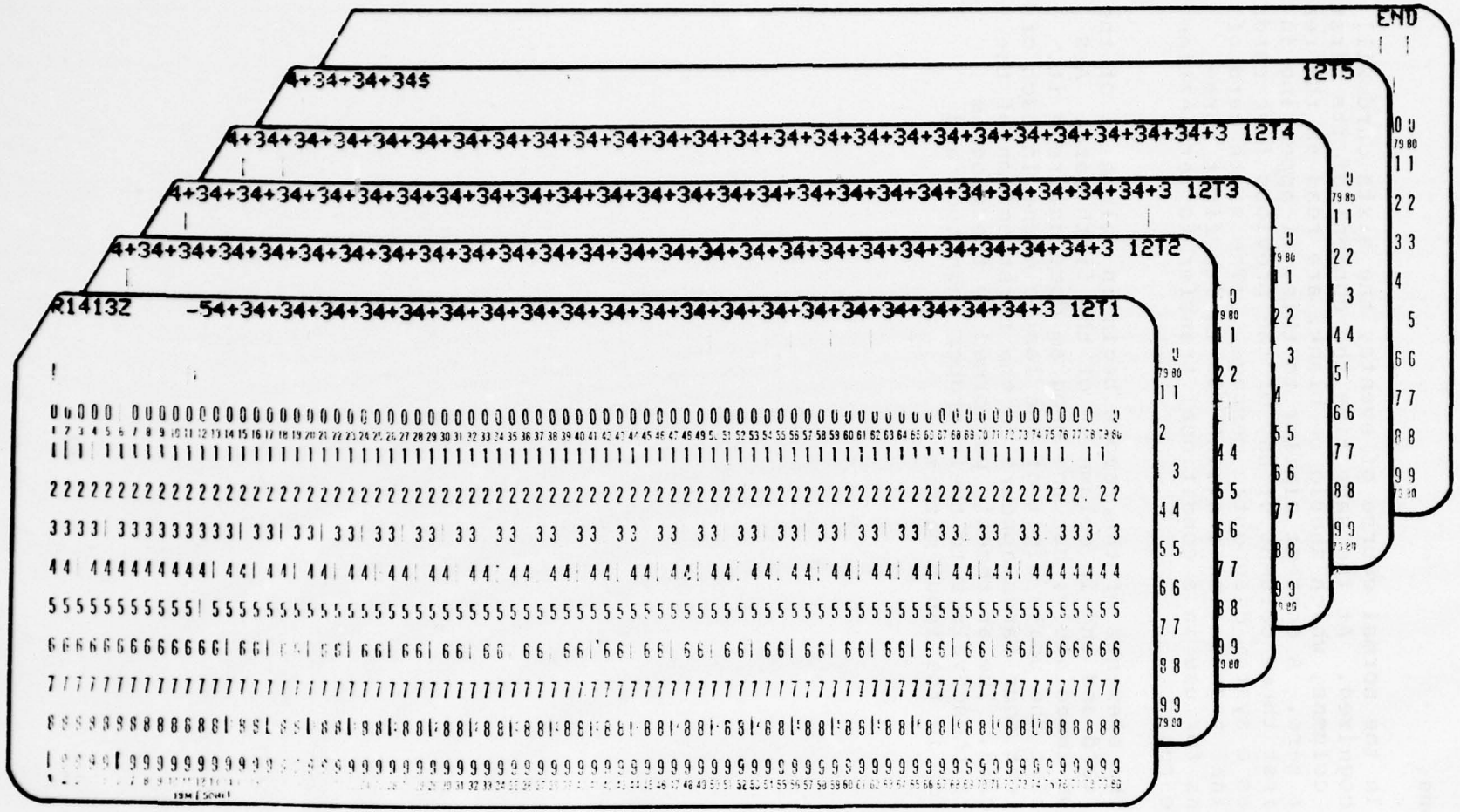


FIGURE 7

TRANSACTION DECK

new one.

In the normal course of events, the sixth card will be recognized. At the same time the content of its first three columns, which should be blank, are read as the real number zero. A content similar to this, if appearing in the first three columns of any of the previous five cards invokes a system diagnostic message. (The sixth card of the final transaction contains 999 in its first three columns for use in a conditional transfer to termination of the run).

The elements of the record begin in column 11 of the first card and end in column 10 of the fifth card. A $ follows immediately, but the program does not read it. Note that the units digit of the last element in each of the first four cards appears in the first column of the next card. The arithmetic performed by the program adjusts these values to their proper magnitude and position in the input matrix.